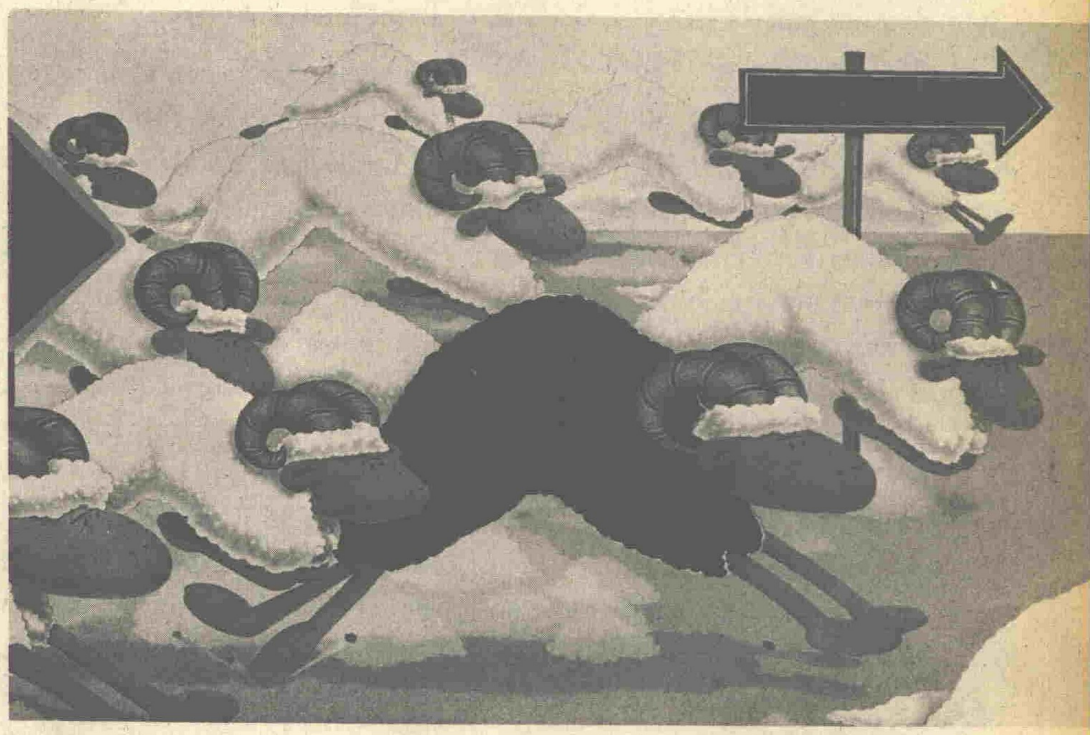
ซอฟต์แวร์
กิติพงษ์ ฤทธิบุตร

Integrated Debugger
นักสืบ Turbo C (3)

วิธีขจัดบักในโปรแกรม และแก้ไขให้ถูกต้อง ไม่ใช่เรื่องยากอีกแล้ว ค้นหาเทคนิคนี้ได้ใน Integrated Debugger



คราวนี้เราจะคุยกันต่อจากที่ติดค้างกันไว้ในตอน 2 ซึ่งว่ากันถึงเรื่องของ Integrated Debugger เกี่ยวกับวิธีใช้งานและทำความรู้จักกับคำสั่งต่าง ๆ ของดีบักเกอร์ตัวนี้ รวมถึงการลองหาบักในโปรแกรมตัวอย่างกันบ้างแล้ว สำหรับตอนนี้ผู้เขียนจะพาผู้อ่านไปพบกับบักพร้อมการแก้ไขการทำงานของโปรแกรมให้ถูกต้อง เนื้อหาส่วนใหญ่จะเป็นเรื่องของการใช้คำสั่งของดีบักเกอร์เพิ่มเติม โดยนำเทคนิคบางอย่างมาใช้เพื่อช่วยค้นหาบักในโปรแกรม และปิดท้ายรายการด้วย "สิ่งละอัน พันละน้อย" สาระน่ารู้สำหรับการใช้งานดีบักเกอร์เพื่อหาข้อผิดพลาดในโปรแกรมที่เขียนขึ้นเอง

มาแก้บักกัน

ขณะนี้ได้เวลาเริ่มตรวจสอบบักในโปรแกรม WORDCNT และแก้ไขการทำงานให้ถูกต้องดีกว่า ผู้เขียนเคยกล่าวถึงในตอนต้นแล้วว่า สาเหตุที่เกิดบักก็คือการทำงานของฟังก์ชัน 'wordlen' ถ้าเรายังอยู่ระหว่างการทำงานของดีบักเกอร์ให้ยกเลิกเสียก่อนด้วยการใช้คำสั่ง Run/Program reset (หรือใช้ฮอตคีย์ <Ctrl> <F2>) จะส่งผลให้เทอร์โบซีคืนหน่วยความจำที่โปรแกรม WORDCNT ใช้ให้กับระบบและปิดไฟล์ที่เกี่ยวข้องทั้งหมด แล้วหยุดการรันโปรแกรม WORDCNT เมื่อออกจาก Integrated Environment ให้เริ่มต้นใหม่โดยเลือกโปรเจ็คต์ไฟล์ชื่อ WORDCNT.PRJ และกำหนด breakpoint ในโปรแกรมตรงบรรทัดที่มีข้อความ

```
While (fgets(...
```

แล้วเลือกใช้คำสั่ง Run/Run เพื่อเริ่มงานดีบักซ้ำอีกครั้ง ซึ่งผลที่ได้จากการรันจะเหมือนเดิมคือ เทอร์โบซีจะยอมให้โปรแกรมรันไปได้จนถึง breakpoint ที่กำหนดไว้

เมื่อโปรแกรมรันถึงช่วง input file name ให้พิมพ์

```
WORDCNT.DAT <Enter>
```

เลือกคำสั่ง Run/Step over เพื่อรันโปรแกรม WORDCNT ทีละคำสั่งจนถึงฟังก์ชัน 'wordlen' ให้เลือกคำสั่ง Run/Trace into (หรือใช้ฮอตคีย์ <F7>)

เพื่อให้ดีบักเกอร์ตรวจสอบการทำงานภายในฟังก์ชัน 'wordlen' ซึ่งคำสั่งนี้จะรันโปรแกรมเพียงครั้งละหนึ่งบรรทัด เช่นเดียวกันกับคำสั่ง Run/Step over แต่จะต่างกันตรงที่ Run/Trace into ตรวจสอบภายในฟังก์ชันที่ถูกเรียกใช้ด้วย และในกรณีนี้ระบบจะวางตำแหน่งของ execution bar ไว้ที่คำสั่งแรกของฟังก์ชัน 'wordlen'

สิ่งที่จะต้องกระทำต่อไปคือศึกษาขั้นตอนการทำงานของฟังก์ชัน เราจะเห็นว่ามีการเรียกใช้พารามิเตอร์หนึ่งตัว คือพอยน์เตอร์ที่จะชี้คำในไลน์บัฟเฟอร์ ใช้ชื่อว่า 'wordptr' การทำงานของฟังก์ชันจะเป็นการกำหนดค่าของ 'wordptr' ให้กับตัวแปร (local variable) ชื่อ wordlimit แล้วจึงเข้าไปทำงานใน while loop เพื่อเพิ่มค่าให้กับ wordlimit จนกว่าจะชี้ช่องว่างที่แสดงว่าจบคำหรือ null character ในกรณีที่จบบรรทัด ค่าที่ฟังก์ชันส่งคืนมาให้คือค่าแตกต่างระหว่าง wordlimit และ wordptr เท่ากับความยาวของคำ

วิธีการที่ช่วยตรวจสอบว่าฟังก์ชัน 'wordlen' ทำงานตรงตามความต้องการหรือไม่ โดยการรันโปรแกรมต่อไปอีกสองบรรทัด เพื่อให้โปรแกรมประมวลผลจนถึง assignment statement ในกรณีนี้การสั่งรันอาจใช้คำสั่ง Run/Trace into หรือ Run/Step over จะได้ผลลัพธ์เช่นเดียวกัน เพราะไม่มีการเรียกใช้ฟังก์ชันอื่นต่อไป จากนั้นจะเป็นการประมวลผลสตริงที่ wordlimit ชี้อยู่ คือบรรทัดแรกของอินพุตนั่นเอง หากโปรแกรมทำงานถูกต้อง เมื่อเราสั่งรันคำสั่ง while แล้ว execution bar ควรจะย้ายไปบรรทัดต่อไปซึ่งจะเป็นคำสั่งเพิ่มค่าของ wordlimit แต่ในกรณีของเราจะเลื่อนไปยังคำสั่งถัดไป คือคำสั่งรีเทอร์น และนี่อาจจะเป็นบักที่เรากำลังมองหา เมื่อถึงจุดนี้เรากดแป้น <Esc> สองครั้งจะเป็นการยกเลิกการประมวลผล และกลับไปสู่ TC Editor แต่ก็สามารถใช้แป้น <F10> หรือฮอตคีย์ของระบบแทนได้

สิ่งที่เราจะต้องทำต่อไปคือ ลองพิจารณากันว่าทำไมถึงไม่มีการเพิ่มค่าให้กับ wordlimit ส่งผลให้ความแตกต่างระหว่าง Wordlimit และ wordptr มีค่าเท่ากับ 0 ผลที่ตามมาคือฟังก์ชัน 'wordlen' จะคืนค่า 0 กลับมาให้ ซึ่งเป็นผลลัพธ์ที่ผิดจากความต้องการและเป็นบักที่เรากำลังหานั่นเอง หมายความว่าเราได้ประมวลความคิดในสิ่งที่เราคิดว่ามีข้อผิดพลาดบางอย่างในฟังก์ชัน 'wordlen' ให้เป็นเพียงสิ่งผิดพลาดที่เกิดขึ้นใน 'while' statement ความจริงเราอาจจะใช้วิธีหาข้อผิดพลาดด้วยตาจากซอร์สโค้ด แต่ตอนนี้เรากำลังใช้งานดีบักเกอร์ คราวนี้เราจะลองประมวลผล expression ใน while statement กันบ้าง เราจะพบว่าค่าของ *wordlimit คือ 'T' (รหัสแอสกี) และค่าของ *wordlimit !=SPACE เป็น 1 (มีค่าเป็นจริง) ด้วย (สำหรับค่าของ SPACE ได้ถูก define ไว้ให้มีค่าเป็น 0x20 หมายถึง ' ' (ช่องว่าง) ด้วยคำสั่ง #define... แล้ว) ดังนั้นผลของ expression ที่ควรจะได้เป็น true แต่กลับปรากฏว่าออกมาเป็น false (มีค่าเป็นเท็จ) แสดงว่าความผิดพลาดในการทำงานเกิดขึ้นที่ expression นี้เอง และตัวที่สร้างปัญหาก็คือตัวโอเปอเรเตอร์นั่นเอง

การใช้ operator '&' ในความหมายนี้ไม่ถูกต้อง เพราะ '&' เป็น

คำสั่ง 'bitwise and' ของภาษาซี จะทำการ 'AND' แต่ละบิตของโอเปอแรนด์ตัวหนึ่งกับบิตที่เกี่ยวข้องกันของโอเปอแรนด์อีกตัวหนึ่ง

ในกรณีนี้เราไม่ต้องการให้โปรแกรมทำงานเช่นนี้ แต่ต้องการทราบว่า expression ของเรามีค่าเป็นจริงหรือเท็จเท่านั้น จะเห็นว่า *wordlimit != SPACE แต่ของเรามีค่าเพียง 0 กับ 1 (False หรือ True) จะทำให้ expression ใน *wordlimit & *wordlimit 1- SPACE มีค่าเป็น 0 เสมอ และถ้า *wordlimit จะเป็นเลขคู่ (0) คือเป็นเท็จ (false) หมายความว่าจะมีอยู่กรณีเดียวที่ expression นี้จะให้ค่าเป็นจริงคือเมื่อโอเปอแรนด์ทั้งสองตัวมีค่าเป็น 1 (true) ทั้งคู่ และตัวโอเปอเรเตอร์ที่เราต้องการก็คือ '&&' (logical AND) ไม่ใช่ '&' (bitwise AND)

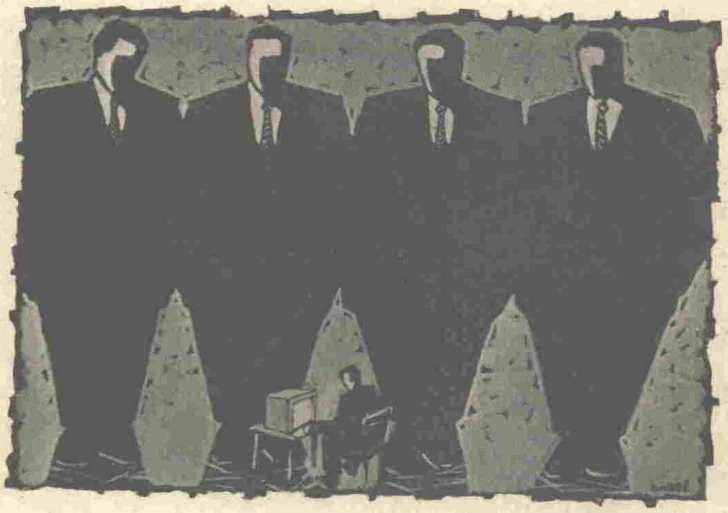
ก็เป็นอันว่าเรารู้ข้อผิดพลาดของโปรแกรมแล้ว ต่อไปเราต้องแก้ไขให้ถูกต้อง (เปลี่ยนเครื่องหมายจาก & เป็น &&) แล้วทำการเก็บ (save) โปรแกรมไว้โดยการกดแป้น <F2> (เก็บไฟล์ที่แก้ไขแล้วลงดิสก์) ขั้นต่อมา

ก็ใช้คำสั่ง Run/Run อีกที ระบบจะถามว่าต้องการให้สร้างโปรแกรมใหม่หรือไม่ เพราะเราได้ทำการแก้ไขชุดเก่าไปแล้ว ฉะนั้นเราต้องพิมพ์คำว่า 'Y' ระบบจะทำการแปลและลิงก์เพื่อสร้าง WORDCNT.EXE ชุดใหม่ขึ้นมาได้ ซึ่งโปรแกรมใหม่นี้จะทำลายบักไป 1 ตัว ขั้นตอนต่อไปต้องพยายามหาบักที่มีอยู่ในส่วนอื่นของโปรแกรม เรามาสรุปกันก่อนดีกว่าว่าขั้นตอนที่เราทำมาตั้งแต่รู้ว่ามีบักอยู่ในส่วนไหนของโปรแกรม จนกระทั่งทำการแก้ไขและสร้างโปรแกรม WORDCNT ขึ้นใหม่มีอะไรบ้าง

เราเริ่มจากการยกเลิกการแก้ไขโปรแกรมในช่วงแรกที่ได้ทำไว้ด้วยการใช้คำสั่ง Run/Program reset (หรือใช้ <Ctrl> <F2> ที่เป็นฮอตคีย์) แล้วรันโปรแกรม WORDCNT ด้วยคำสั่ง Run/Run (หรือ <Ctrl> <F9>) จนถึงการเรียกใช้ฟังก์ชัน 'wordlen' ที่เราได้กำหนด breakpoint ไว้ตรงนี้ จากนั้นจะทำการตรวจสอบการทำงานของฟังก์ชันนี้ โดยใช้คำสั่ง Run/Trace into และเราจะพบบักที่นี่ เมื่อแก้ไขและเก็บซอร์สไฟล์ที่อัพเดตโปรแกรมแล้วเพื่อพร้อมที่จะใช้งานต่อไปโดยใช้คำสั่ง Run/Run ระบบจะทำการสร้างไฟล์ 'WORDCNT.EXE' ขึ้นใหม่ หมายความว่า เมื่อถึงตรงนี้เราสามารถใช้งาน Integrated Debugger ได้ในระดับหนึ่ง สามารถหาบักและแก้ไขได้สำเร็จ

เรื่องน่ารู้ของ breakpoint

ก่อนที่จะว่ากันต่อถึงคุณสมบัติข้ออื่น ๆ ของดีบักเกอร์ตัวนี้ ผู้เขียนขอเข้าเรื่อง breakpoint เพื่อให้เราทำความ



รู้จักและใช้งานได้ดียิ่งขึ้น ซึ่งสิ่งแรกคือ ตำแหน่ง breakpoint ที่กำหนดไว้ในซอร์สโค้ดนั้นยังคงอยู่ในตำแหน่งเดิม ถึงแม้ว่าเราจะออกจากการทำงานของดีบักเกอร์กี่ครั้งก็ตาม จะเห็นได้จากการใช้คำสั่ง Run/Run โปรแกรม WORDCNT หลาย ๆ ครั้ง นอกจากนี้เทอร์โบซียังช่วยปรับหรือคอยดูแลให้ breakpoint อยู่ในตำแหน่งที่ถูกต้องเสมอ ไม่ว่าเราจะทำการแก้ไขซอร์สโปรแกรมอย่างไร เช่น ในกรณีนี้โปรแกรมจะหยุดรันที่บรรทัดคำสั่ง

```
'While (fgets (.....
```

อย่างไรก็ตาม ตำแหน่งของ breakpoint อาจเสียไป (เทอร์โบซีจำไม่ได้ว่าอยู่ที่ไหน ทำให้การจำตำแหน่งผิดพลาด) หรือยกเลิกได้ในกรณีต่อไปนี้คือ

– ออกจากการทำงานของ Integrated Environment

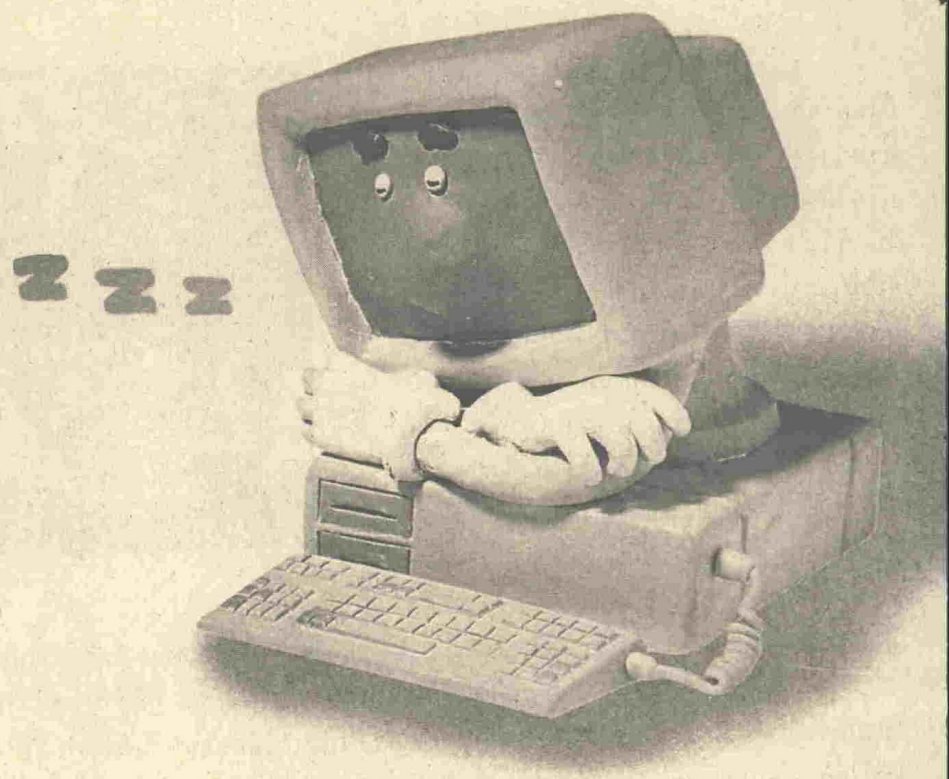
– ลบบรรทัดที่กำหนดไว้เป็นตำแหน่งของ breakpoint ออก

– ยกเลิกตำแหน่งที่เป็น breakpoint ด้วยคำสั่ง Break/Watch/Clear all breakpoints

ส่วนสาเหตุหนึ่งที่ทำให้เทอร์โบซีกำหนดตำแหน่งของ breakpoint ผิดพลาดจากที่กำหนดไว้เดิมนั้นมีอยู่ 2 กรณีคือ

1. ถ้าเราแก้ไขไฟล์ที่มีการกำหนดตำแหน่งของ breakpoint ไว้ แล้วยกเลิกโดยไม่ได้เก็บไฟล์ที่แก้ไข ทำให้เทอร์โบซีไม่สามารถจดจำตำแหน่งของ breakpoint ที่ถูกต้องได้ และจะทำให้ตำแหน่ง Breakpoint ที่มีอยู่คลาดเคลื่อนไปจากเดิม เพราะฉะนั้นในการยกเลิกไฟล์ที่กำลังแก้ไขอยู่ (ยกเลิกการแก้ไขที่ทำทั้งหมด) ก็ควรจะยกเลิกตำแหน่งของ Breakpoint ทั้งหมดที่มีอยู่ด้วยโดยใช้คำสั่ง Break/Watch/Clear all breakpoints แล้วค่อยทำการกำหนดขึ้นใหม่ทีหลัง สำหรับคำสั่งที่ว่านี้จะมีผลให้เทอร์โบซีลบตำแหน่งของ breakpoint ทุกตำแหน่งในโปรแกรม ไม่ใช่เฉพาะตำแหน่งที่อยู่ในซอร์สไฟล์ที่กำลังแก้ไขอยู่เท่านั้น

2. ถ้าทำการแก้ไขไฟล์ แล้วเข้าสู่การดีบักโปรแกรม หรือเริ่มทำการดีบักใหม่โดยที่ไม่ได้แปลและลิงก์ใหม่อีกครั้ง ในกรณีนี้ตำแหน่งของ breakpoint ที่กำหนดไว้ใหม่จะยังคงอยู่ใน



ตำแหน่งที่ถูกต้องตามที่ได้กำหนดไว้ใหม่ แต่ตำแหน่งที่โปรแกรมหยุดทำงานเมื่อสั่งรันนั้นอาจจะไม่ตรงกันกับตำแหน่งของ breakpoint เพราะดีบักเกอร์ยังรันโปรแกรมชุดเก่า และ execution bar จะอยู่ในตำแหน่งที่ผิดด้วย อย่างไรก็ตาม กรณีนี้มักจะไม่เกิดขึ้นเพราะเทอร์โบซีจะแสดงข้อความเตือนผู้ใช้คือ

"Source modified, rebuild ?"

เมื่อผู้ใช้พยายามที่จะทำการดีบักต่อหรือเริ่มใหม่อีกครั้งโดยยังไม่ได้ทำการแปลใหม่

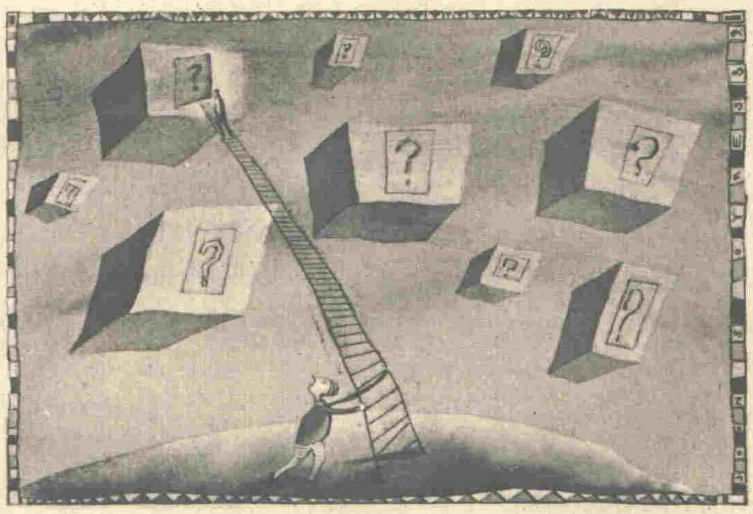
ปกติแล้วการกำหนดตำแหน่งของ breakpoint นั้นจะกำหนดบนบรรทัดที่เป็น executable statement ถ้าเรากำหนดบนบรรทัดที่ไม่ใช่ executable statement เช่น บรรทัดที่เป็น comment หรือบรรทัดว่างแล้ว เทอร์โบซีจะเตือนให้รู้ก่อนที่จะรันโปรแกรม เพราะในขณะที่คอมพายล์โปรแกรม เทอร์โบซีจะรู้ว่าบรรทัดไหนของโปรแกรมที่ใช่หรือไม่ใช่ executable statement และจะเตือนหากผู้ใช้พยายามที่จะกำหนดตำแหน่งของ breakpoint ไว้บนบรรทัดเหล่านั้น

นอกจากนี้ยังมีอีกคำสั่งหนึ่งที่ควรให้ความสนใจคือ Break/Watch/View next breakpoint ซึ่งจะทำหน้าที่เลื่อนแถบเคอร์เซอร์ไปยังตำแหน่งของ breakpoint จุดต่อไปเป็นการเลื่อนไปที่ breakpoint ที่ต้องการให้หยุดครั้งต่อไปใน current project ไม่ใช่ตำแหน่งต่อไปที่จะทำการประมวลผลเมื่อเราใช้คำสั่งรัน ดังนั้นเราสามารถใช้คำสั่งนี้ค้นหาตำแหน่งของ breakpoint จุดต่อไปในกรณีที่ต้องการจะยกเลิกตำแหน่งของ breakpoint บางแห่ง

ใช้งาน Watch window

บน TC screen ซึ่งเป็นเมนูหลักของเทอร์โบซีในแบบ Integrated Environment นั้นจะแบ่งจอภาพออกเป็นสองส่วนคือ ส่วนของ Edit window คือส่วนที่ระบบทำหน้าที่ดึงโปรแกรมที่กำลังใช้งานขึ้นมาแสดงให้เราดู สามารถแก้ไขหรือพัฒนาโปรแกรมได้ภายในส่วนของวินโดว์นี้ อีกส่วนหนึ่งคือ Watch window จะอยู่บริเวณตอนล่างของจอภาพใช้ในการติดต่อ สื่อความหรือให้ข้อมูลบางอย่างเกี่ยวกับการทำงานของโปรแกรมแก่ผู้ใช้ มาดูกันว่าเราสามารถใช้ประโยชน์จากวินโดว์นี้ได้อย่างไรบ้าง

บางคนอาจจะสงสัยว่าโปรแกรม WORDCNT ที่แก้ไขบักออกไปแล้วหนึ่งแห่งสามารถทำงานได้ถูกต้องตรงความต้องการหรือไม่ ถ้าเราออกจาก Integrated Environment แล้ว หรือรันโปรแกรมอื่นอยู่ก็เริ่มดีบักใหม่ด้วยการเรียกใช้ TC อีกครั้ง และใช้คำสั่ง Run/Run โปรแกรมจะรันต่อไปจนถึงคำสั่งที่มีการเรียกใช้ wordlen ที่เป็น breakpoint จุดแรก ลองใช้คำสั่ง Run/Trace into เพื่อตรวจสอบการทำงานของ wordlen หากคราวนี้เราได้คำตอบเป็น 2 แสดงว่าการทำงานของโปรแกรมเป็นไปด้วยความเรียบร้อย จุดต่อไปที่เราควรให้ความสนใจเมื่อเราต้องการตรวจสอบความถูกต้องในการทำงานของโปรแกรมก็คือ while loop อีกชั้นหนึ่ง (inner while loop) ในฟังก์ชัน 'main' เราใช้วิธีตรวจสอบการทำงานของ main ( ) จนหมดข้อมูลในบรรทัดแรกของ input file ระบบจะเลิกการทำงานภายในลูป เราจะคอยดูว่าการทำงานแต่ละขั้นตอนนั้นถูกต้องหรือไม่ และเมื่อทำงานเรียบร้อยแล้วการทำงาน



ของโปรแกรมยังเป็นไปตามปกติหรือไม่ด้วย ซึ่งงานเหล่านี้เราสามารถจะใช้ประโยชน์จาก Watch window ได้ เพราะ Watch window จะแสดง expression ต่าง ๆ รวมทั้งค่าที่ได้ให้เห็นซึ่งทุกครั้งที่ดีบักเกอร์หยุดทำงานชั่วคราว (halt) โปรแกรมจะประมวลผลแต่ละ expression ที่อยู่ใน Watch window ใหม่ พร้อมทั้งแสดงผลลัพธ์ด้วย หมายความว่าเราสามารถตรวจสอบผลการทำงานของ expression ได้จาก Watch window ในขณะรันโปรแกรม

สิ่งที่เราจะต้องทำต่อไปคือสร้าง Watch expression สำหรับตรวจสอบข้อมูลทุก ๆ ตัวที่มีความสำคัญต่อการทำงานภายในลูปนั้น ๆ อันประกอบด้วยตัวแปร i กับสตริงที่เป็นตัวเริ่มต้นที่ wordptr ชี้ และข้อมูลสมาชิกของ wordlencnt คราวนี้เราลองกลับมาดูที่ Watch window ที่อยู่แห่งเดียวกันกับ Message window แต่ Message window จะทำงานเมื่อเริ่มใช้คำสั่งคอมพายล์และลิงก์ ซึ่งตามปกติวินโดว์นี้จะมีขนาดความกว้างสำหรับข้อมูลข่าวสารเพียงบรรทัดเดียว (ถ้ามองไม่เห็นแสดงว่ามีการเรียกใช้คำสั่ง Zoom ทำให้ Edit window ใช้พื้นที่เต็มจอภาพ แต่เราสามารถเรียกวินโดว์นี้กลับคืนมาได้อีกด้วยการกดแป้น <F5> ที่เป็นบรรทัดว่างอยู่ ขั้นต่อไปให้เรียกคำสั่ง Break/Watch/Add watch (หรือใช้ฮอตคีย์ของคำสั่งคือ <Ctrl> <F7>) ดีบักเกอร์จะเปิดวินโดว์เตรียมพร้อมที่จะรับคำสั่งจากผู้ใช้ต่อไป ปกติแล้วคำสั่งนี้จะมี default expression อยู่ใน Edit window เช่นเดียวกับคำสั่ง Debug/Evaluate คือจะแสดงคำที่เคอร์เซอร์ทับอยู่ แต่ถ้าไม่มีให้พิมพ์ 'i' แล้วกด <Enter> ซึ่งดีบักเกอร์ก็จะเพิ่ม expression 'i' และค่าของมันใน Watch window เรายังมี expression อีกหลายชุดที่จะต้องใส่เข้าไปในวินโดว์นี้ เราจึงต้องทำขั้นตอนต่าง ๆ ซ้ำเพื่อเติม expression ที่ต้องการเข้าไปอีก ได้แก่ wordptr และ wordlencnt [1] จนถึง wordlencnt [4] ทุกครั้งที่เพิ่ม expression เข้าไป Watch window จะขยายตัวเพิ่มขึ้นตามลำดับ เมื่อกำหนด expression ต่าง ๆ ครบตามต้องการแล้วก็เริ่มลงมือตรวจสอบการทำงานของ while loop ได้เรื่อย ๆ จนกว่าโปรแกรม WORDCNT จะประมวลผลข้อมูลในบรรทัดแรกของ input file หมด และในขณะที่โปรแกรมทำงานเราสามารถมองเห็น WORDCNT เลื่อน wordptr ไปทีละคำ โดยจะเพิ่มค่าของ wordlencnt ด้วย เราจะให้โปรแกรมหยุดทำงานเมื่อทำไปจนจบบรรทัดของ input และ execution bar เคลื่อนออกมาอยู่นอกลูปแล้ว ซึ่งเราจะเห็นว่าการทำงานของโปรแกรมนั้นยังคงเป็นไปอย่างถูกต้อง

ว่าด้วยเรื่อง

Watch expression

การใช้ Watch window นั้น นอกจากเราจะสามารถใส่ expression เข้าไปได้ตามต้องการแล้ว เรายังสามารถเข้าไปแก้ไข expression เหล่านั้นได้เช่นกัน ไม่ว่าจะเป็นการแก้ไข ลบทิ้ง หรือเพิ่มเติม ซึ่งวิธีการใช้งานจะเริ่มจากการเรียกใช้ Watch window ในกรณีที่เราอยู่ในเมนซิสเต็มส์ใช้ฟังก์ชันคีย์ <F10> สั่งให้ระบบกลับไปยังแอคทีฟวินโดว์ที่เราใช้งานอยู่ก่อนหน้านี้ (หมายถึง Edit window) แล้วจึงกดแป้น <F6> เพื่อเปลี่ยนให้ Watch window เป็นแอคทีฟวินโดว์แทน ส่งผลให้ Watch editor แสดงแถบสว่างบน expression ที่สามารถแก้ไขได้ หรือเลื่อนแถบสว่างไปยัง expression ที่ต้องการด้วยการกดแป้น <Home>, <End>, <Up arrow> และ <Down arrow>

วิธีการแก้ไข Watch expression ให้เลื่อนแถบสว่างไปที่ wordlencnt [4] เรียกใช้คำสั่ง Break/Watch/Edit watch หรือกดแป้น <Enter> สั่งให้ดีบักเกอร์เปิดวินโดว์ที่มี Watch expression ที่เลือกไว้ เตรียมพร้อมให้เราได้แก้ไข ขั้นตอนต่อไปเราต้องแก้ไขค่าตัวแปร (index) ของอะเรย์จาก 4 เป็น 6 แล้วกด <Enter> ดีบักเกอร์จะเปลี่ยนค่า Watch expression ใน Watch window ให้เป็นค่าใหม่ตามที่

แก้ไขไว้ เลื่อนแถบสว่างไปที่ word-lencnt [3] และใช้คำสั่ง Break/Watch/Delete watch (หรือใช้ฮอตคีย์ <Ctrl><Y> หรือแป้น <Del> ก็ได้) จะทำให้ดีบักเกอร์ลบ Watch expression นั้นออกไปจาก Watch window

สำหรับการลบ Watch expression เราอาจลบทุก expression ทิ้งไปได้ในครั้งเดียวโดยใช้คำสั่ง Break/Watch/Remove all watches และถ้าใช้คำสั่งนี้ก็ไม่มีความจำเป็นที่จะต้องสั่งให้ Watch window เป็นแอคทีฟวินโดว์ก่อน เมื่อเราลองใช้คำสั่งนี้จะเห็นว่าดีบักเกอร์ได้ลบ Watch expression ทั้งหมดทิ้งไป เหลือแต่ Watch window เปล่าที่มีที่ว่างเพียงบรรทัดเดียวเช่นเดิม ทีนี้เราลองกด <F6> จะเห็นว่าระบบได้เปลี่ยนให้ Edit window เป็นแอคทีฟวินโดว์แทน สำหรับแป้น <F6> นี้จะใช้เป็นสวิตช์สำหรับเปลี่ยนแอคทีฟวินโดว์ระหว่าง Edit window และ Message/Watch window

ควบคุม Window

คำสั่งอีกตัวหนึ่งสำหรับควบคุมขนาดของแอคทีฟวินโดว์ให้ใช้พื้นที่เต็มจอภาพก็คือ Zoom โดยการกดแป้น <F5> ซึ่งใช้เป็นสวิตช์สำหรับเปลี่ยนขนาดของวินโดว์ระหว่างแบบเต็มจอกับแบ่งจอ นอกจากนี้ยังมีแป้น <Alt><F5> หรือฮอตคีย์ของคำสั่ง Run/User screen เพื่อสั่งให้ระบบแสดง execution screen และกลับสู่จอภาพปกติได้ด้วยการกดแป้นใด ๆ ก็ได้ อีกคำสั่งหนึ่งก็คือการกดแป้น <Alt><F6> เพื่อให้ระบบเปลี่ยนข้อมูลต่าง ๆ (contents) ภายในวินโดว์ คำสั่งนี้จะทำงานสองลักษณะคือ

- เมื่อ Edit window เป็นแอคทีฟวินโดว์ การใช้แป้นนี้จะมีผลให้ไฟล์ที่ถูกเรียกใช้ (loaded) ก่อนหน้าไฟล์ที่กำลังใช้งานอยู่ (current file) ถูกเรียกขึ้นมาใช้อีกครั้ง (reloaded)
- ในกรณีที่ Watch window หรือ Message window เป็นแอคทีฟวินโดว์ การกดแป้น <Alt><F6> จะเป็นการสั่งให้สลับการใช้วินโดว์ระหว่าง Watch editor และ Message tracking window

กลับมาที่ Watch window อีกครั้ง เพราะยังมีบางเรื่องที่เรายังไม่ได้พูดถึง นั่นคือ การขยายขนาดของ Watch window เมื่อเราเพิ่ม expression เข้าไปจนเป็นผลให้ขนาดของวินโดว์เพิ่มขึ้นเรื่อย ๆ นั้น Watch window จะขยายได้เต็มที่ประมาณครึ่งจอภาพ แต่ถ้ายังมีการเพิ่ม expression อีกเมื่อวินโดว์ขยายตัวเต็มที่แล้วก็จะทำให้เกิดการเลื่อน expression บางส่วนพ้นจากหน้าจอของวินโดว์ ซึ่งส่วนที่ขาดหายไปเราสามารถเรียกดูได้โดยใช้แป้น <PgUp>, <PgDn>, <Up arrow> และ <Down arrow> และในกรณีที่ Watch expression มีขนาดใหญ่เกินกว่าขนาดวินโดว์ เราสามารถดูส่วนที่ขาดหายไปได้โดยการเลื่อนจอภาพด้วยแป้น <Home>, <End>, <Left arrow> และ <Right arrow>

บทสรุป (3)

หลังจากติดตามบทความเรื่องนี้มาจนถึงตอนที่ 3 ผู้อ่านคงพอมองเห็นคุณประโยชน์ของ Integrated Turbo C Debugger เช่นเดียวกับผู้เขียน เพราะหลาย ๆ คนที่ชอบเขียนโปรแกรมเองมักเจอะเจอบักซุกซ่อนอยู่ในโปรแกรม เป็นสาเหตุให้โปรแกรมทำงานผิดพลาด จนโปรแกรมเมอร์ต้องปวดหัวกันบ่อย ๆ กว่าจะหาตัวได้ ซึ่งในตอนนี้ผู้เขียนได้นำเสนอวิธีหาบัก รวมทั้งแก้ไขการทำงานของโปรแกรมให้ถูกต้องใน while loop ของฟังก์ชัน 'main( )' โดยใช้คำสั่งต่าง ๆ ที่ดีบักเกอร์ตัวนี้เตรียมไว้ให้อย่างพร้อมมูล สำหรับคราวหน้าจะพูดถึงแนวทางการค้นหาบักใน print loop บ้าง แถมท้ายด้วยสาระน่ารู้ในการใช้งานโปรแกรมตัวนี้เพิ่มเติมอีกเล็กน้อย

ติดตามกันใหม่นะครับ

หมายเหตุ ผู้สนใจสามารถหาซอฟต์แวร์นี้ได้ที่ร้านบุ๊คเซสท์ สยามสแควร์ ซอย 2